Базові поняття про предмет.

Структура і компоненти програмного забезпечення

Програмне забезпечення можна розділити на три групи:

- системне програмне забезпечення;
- прикладне програмне забезпечення;
- інструментальне програмне забезпечення.

Системне програмне забезпечення управляє роботою всіх програмних і апаратних компонент комп'ютера, забезпечує інтерфейс з користувачем, управляє взаємодією компонента мережі з іншими компонентами, підключеними до нього по каналах зв'язку.

Основним компонентом системного програмного забезпечення комп'ютера є **операційна система** (ОС).

Прикладні програми (додатки) і комплекси (пакетів) програм використовуються для кінцевих пристроїв, в тому числі комп'ютерів і мобільних телефонів (редактори тексту, системи управління базами даних СУБД, ігрові програми і т.д.)

Інструментальне програмне забезпечення використовується для створення нових програм.

Сучасні інструментальні засоби, звані також **системами програмування**, включають потужні і зручні засоби для написання, модифікації і тестування програм, а також включають готові програми (бібліотеки) для реалізації найбільш часто використовуваних операцій.

Java та інші мови програмування.

1956 – **Fortran**, перша мова програмування ***високого рівня***

1972 - **C**, мова ***процедурного програмування***, її базовими конструкціями є ***підпрограми***. Створення системного програмного забезпечення.

1967—1972 напрямок ***об'єктного програмування***, заснованого на концепціях роботи з ***класами*** та ***об'єктами***

1974 - **Pascal**, мова ***структурного програмування***, компіляція програм за рахунок ***віртуальної машини***.

Java та інші мови програмування.

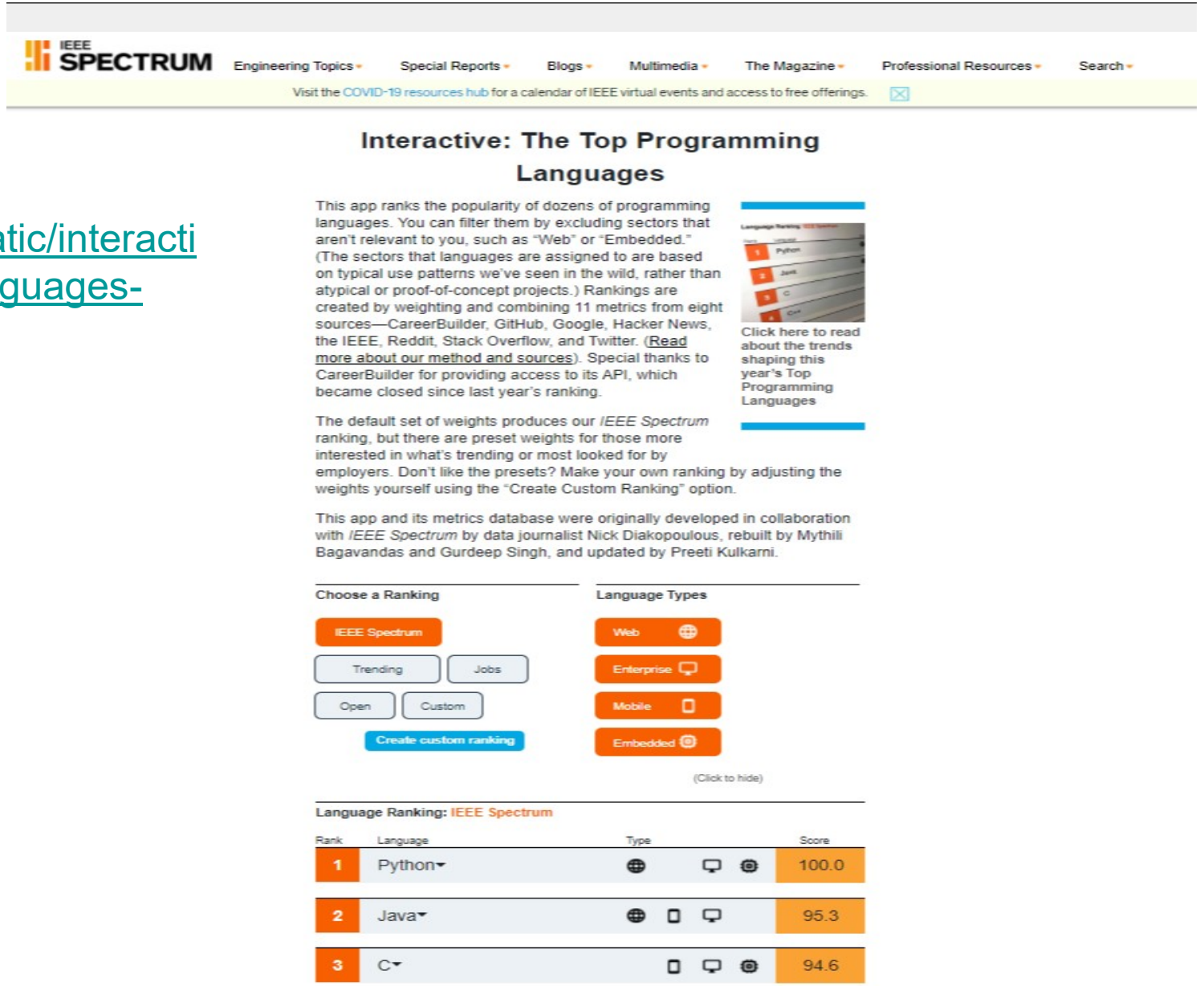
1983 - розширення мови **C**, перший компілятор мови **C++** ,
універсальна мова системного програмування

1995 - **Java**, універсальна мова прикладного програмування,

JavaScript, вузькоспеціалізована мова програмування HTML-документів

2000 - платформа **.NET**, альтернатива платформі Java

https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2020



IEEE SPECTRUM Engineering Topics Special Reports Blogs Multimedia The Magazine Professional Resources Search

Visit the COVID-19 resources hub for a calendar of IEEE virtual events and access to free offerings.

Interactive: The Top Programming Languages

This app ranks the popularity of dozens of programming languages. You can filter them by excluding sectors that aren't relevant to you, such as "Web" or "Embedded." (The sectors that languages are assigned to are based on typical use patterns we've seen in the wild, rather than atypical or proof-of-concept projects.) Rankings are created by weighting and combining 11 metrics from eight sources—CareerBuilder, GitHub, Google, Hacker News, the IEEE, Reddit, Stack Overflow, and Twitter. (Read more about our method and sources). Special thanks to CareerBuilder for providing access to its API, which became closed since last year's ranking.

Click here to read about the trends shaping this year's Top Programming Languages

The default set of weights produces our *IEEE Spectrum* ranking, but there are preset weights for those more interested in what's trending or most looked for by employers. Don't like the presets? Make your own ranking by adjusting the weights yourself using the "Create Custom Ranking" option.

This app and its metrics database were originally developed in collaboration with *IEEE Spectrum* by data journalist Nick Diakopoulous, rebuilt by Mythili Bagavandas and Gurdeep Singh, and updated by Preeti Kulkarni.

Choose a Ranking

IEEE Spectrum | Trending | Jobs | Open | Custom | Create custom ranking

Language Types

Web | Enterprise | Mobile | Embedded

(Click to hide)

Language Ranking: IEEE Spectrum

Rank	Language	Type	Score
1	Python		100.0
2	Java		95.3
3	C		94.6

В основу розробки мови Java було покладено такі основні принципи:

- переносимість;
- об'єктна орієнтація;
- багатопоточність;
- розподіленість;
- безпека.

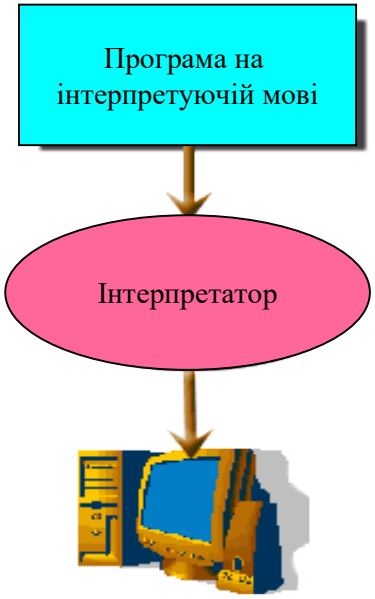
Переносимість

Переносимість називається можливість перенесення програми на інші комп'ютерні платформи та операційні системи

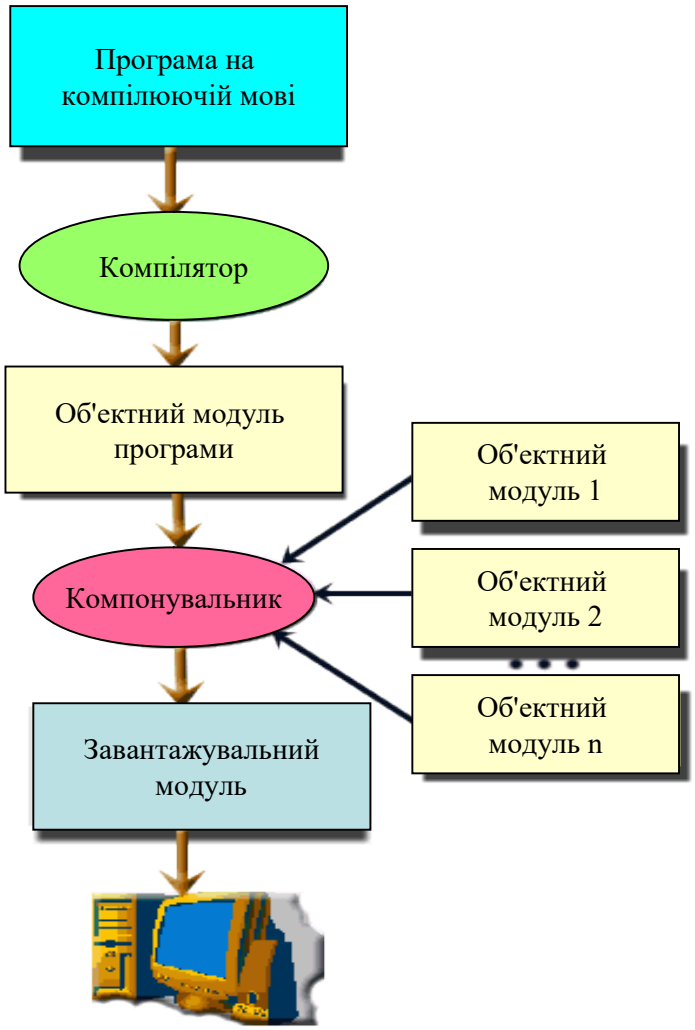
Перетворення програм в машинний код і виконання програми може проводитися одному з двох режимів: інтерпретації та компіляції.

У режимі інтерпретації виконується незалежне перетворення кожної команди програми в машинний код і ця команда або інструкція відразу ж виконується. Інтерпретатор знаходиться в оперативній пам'яті протягом всього часу виконання програми користувача (BASIC, JavaScript, VBScript).

При роботі компілятора програма на вихідній мові спочатку перетвориться в еквівалентну програму на машинній мові - **проміжний машинний код в об'єктному модулі**. Всі об'єктні модулі збираються в єдиний **завантажувальний модуль** за допомогою спеціальної програми - **компонувальника.** Такий модуль може бути завантажений ОС в оперативну пам'ять і виконаний.(C, C++, Pascal).



Виконання програми в режимі інтерпретації



Виконання програми в режимі компіляції

Вихідні програми на мові Java компілюються, але не в машинні команди конкретної комп'ютерної платформи, а в команди **віртуальної машини Java** (JVM, Java Virtual Machine).

Віртуальна машина Java - це сукупність команд разом з програмною системою їх виконання. Команди JVM короткі, більшість з них має довжину 1 байт, тому ці команди називають **байт-кодами** (byte-codes), хоча є команди довжиною 2 і 3 байти. Байт-коди записуються в одному або декількох файлах, можуть зберігатися в зовнішній пам'яті або передаватися по мережі.

Таким чином, **компіляція не залежить від комп'ютерної платформи** і в той же час при виконанні **інтерпретується не текст вихідної програми, а незалежні від комп'ютерної платформи байт-коди**.

Всі стандартні функції Java, що викликаються в програмі, підключаються до неї тільки на етапі виконання, а не включаються в байт-коди (**динамічна компоновка**, dynamic binding) програми. Це сильно зменшує обсяг скомпільованої програми.

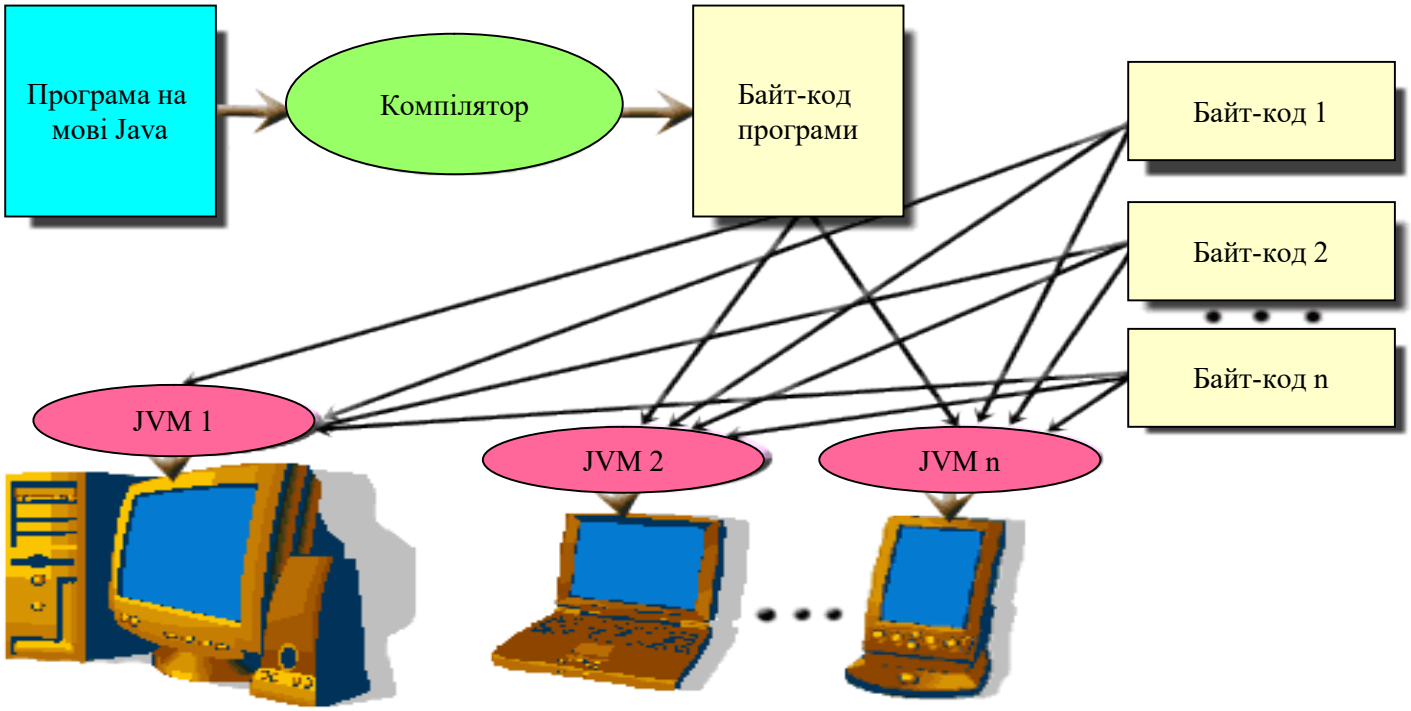


Схема виконання програми на мові Java

Об'єктна орієнтація

Мови програмування по областям застосування можна розділити на дві групи: **мови загального призначення** і **спеціалізовані мови**.

У мовах загального призначення програма традиційно була послідовністю операцій (процедур) над даними різних типів, які реалізують алгоритм вирішення задачі. Тому мови загального призначення називалися також **процедурно-орієнтованими мовами**.

На процедурно-орієнтованій мові можна програмувати завдання будь-яких предметних областей, тому мови програмування загального призначення іноді називають **універсальними мовами програмування**.

Спеціалізовані мови програмування орієнтуються на конкретну предметну область (завдання), тому їх називають також **проблемно-орієнтованими мовами програмування**.

Технологія об'єктно-орієнтованого програмування, що з'явилася на початку 80-х років і була вперше реалізована в мові C, полягає в тому, що програма адаптує себе до мови опису завдання. При **об'єктно-орієнтованому підході** програми розробляються з точки зору залучених до неї об'єктів програмованої предметної області, їх властивостей і поведінки. Об'єктно-орієнтовані мови об'єднують в собі переваги спеціалізованих мов програмування і універсальність мов загального призначення. Переваги об'єктно-орієнтованого підходу особливо яскраво проявляються при написанні великих і складних програм.

Мова Java, так само як і мови C, Object Pascal, Perl, Python і C #, є **об'єктно-орієнтованою мовою**.

Багатопоточність

У сучасних ОС може одночасно виконуватися кілька незалежних завдань (**багатозадачні операційні системи**). З кожним завданням пов'язаний один або кілька процесів, які можуть виконуватися паралельно.
ОС забезпечує захист кожного виконуваного процесу від впливу інших процесів. Високий ступінь ізоляції процесів один від одного дозволяє збільшити стійкість роботи комп'ютера, але при цьому зростають витрати системних ресурсів на запуск і підтримку окремих процесів.

Була запропонована концепція **потоків команд** (**threads** - дослівно «нитки», або **lightweight processes** - спрощені процеси). Потоки команд не захищені один від одного засобами ОС. Їх головна перевага - дуже швидкий запуск.

Java є одним з небагатьох мов програмування, що підтримують режим **роботи з декількома потоками**.

Розподіленість

Розподілене програмування - окремі компоненти програми виконуються в різних вузлах мережі.

Додаток можна розділити на шість функціональних частин:

1. Засоби представлення даних на екрані (GUI, Graphic User Interface);
2. Логіка представлення даних на екрані (вибір з системи меню, вибір елемента зі списку і т. д.);
3. Прикладна логіка - набір правил для прийняття рішень, обчислювальні процедури і операції;
4. Логіка даних - операції з даними, що зберігаються в деякій базі даних, які потрібно виконати для реалізації прикладної логіки;
5. Внутрішні операції бази даних - дії СУБД, що викликаються у відповідь на виконання запитів логіки даних;
6. Файлові операції - стандартні операції над файлами і файлової системою, які зазвичай є функціями операційної системи.

На практиці додаток зазвичай поділяють на дві або три частини. Найбільш поширеною є схема з двох ланок, що розподіляє додаток між двома комп'ютерами. Функціональні частини програми можна розділити між двома комп'ютерами різними способами:

а) **Централізована схема** - комп'ютер або пристрій користувача працює як термінал, що виконує функції представлення даних, всі інші функції передаються від центрального комп'ютера.
Недоліком такої схеми є її недостатня масштабованість і відсутність відмовостійкості.

б) **Схема «файловий сервер»** - на клієнтській машині виконуються всі частини програми, крім файлових операцій. Розподілений додаток в цій схемі мало відрізняється від повністю локального додатка.

в) **Схема, в якій на серверний комп'ютер покладаються функції проведення внутрішніх операцій бази даних і файлових операцій**. Клієнтський комп'ютер при цьому виконує всі функції, специфічні для додатка, а сервер - функції, реалізація яких не залежить від специфіки додатка.

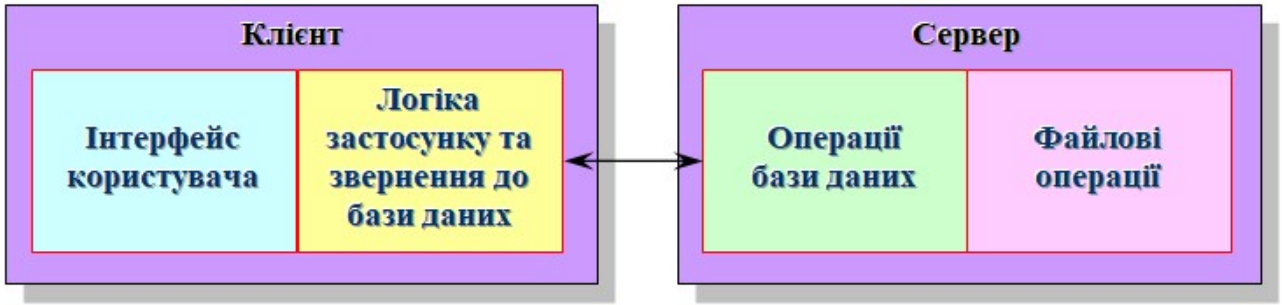
а)


Клієнт
Емуляція терміналу сервера
Сервер
Логіка застосунку та звернення до бази даних
Операції бази даних
Файлові операції

б)


Клієнт
Інтерфейс користувача
Логіка застосунку та звернення до бази даних
Операції бази даних
Сервер
Файлові операції

в)


Клієнт
Інтерфейс користувача
Логіка застосунку та звернення до бази даних
Сервер
Операції бази даних
Файлові операції

Триланкова архітектура дозволяє краще збалансувати навантаження на різні компоненти мережі, а також сприяє подальшій спеціалізації серверів і засобів розробки розподілених додатків.

Триланкові схеми часто застосовуються для централізованої реалізації в мережі деяких загальних для розподілених додатків функцій, відмінних від файлового сервісу і управління базами даних.

Програмні модулі, що виконують такі функції, відносять до класу **middleware**, тобто проміжного шару, розташованому між індивідуальною для кожного додатка логікою і сервером баз даних.

До складу Java включені високорівневі засоби доступу до даних по протоколу HTTP, що використовується для передачі Web-сторінок, а також по транспортним протоколам мережі Internet.

Віддалений виклик процедур в мові Java реалізує технологія виклику віддалених методів **RMI** (**Remote Method Invocation**).

Служба **JNDI** (**Java Naming and Directory Interface** - інтерфейс імен та каталогів Java) дозволяє звертатися до об'єктів мережі з програм, використовуючи при цьому символьні імена об'єктів.

Java містить службу повідомлень - **JMS** (**Java Message Service**), для програмування обміну повідомленнями в мережі, засоби для створення клієнтських додатків електронної пошти, а також засоби для створення мережевих додатків на базі архітектури **CORBA** (**Common Object Request Broker Architecture** - архітектура універсального посередника запитів об'єкта).

Безпека

Програми Java можуть містити потенційно небезпечний код, тому в мові Java передбачені засоби, що забезпечують **безпеку виконання програм**.

Початкова модель безпеки, що забезпечується Java, відома як **модель «пісочниці»,** забезпечувала дуже жорсткі правила виконання програм. Відповідно до цих правил локальна програма мала доступ до ресурсів комп'ютера, а програми, отримані з мережі могли отримати тільки обмежений доступ до ресурсів усередині «пісочниці».

Надалі було введено поняття «**сертифікованого аплету**» (signed applet), який може мати доступ до локальних ресурсів, а потім було введено поняття політики безпеки, реалізованої в Java, починаючи з JDK 1.2.

Політика безпеки на комп'ютері визначає набір дозволів доступних програм від різних джерел і може бути налаштованою користувачем або системним адміністратором.

Лектор:

Старший викладач кафедри Електроніки и комп'ютерної техніки Сумського державного університету

Горячев О. Є.

В лекції використано матеріали авторів:

Шонін В.А.
Монахов В.В.